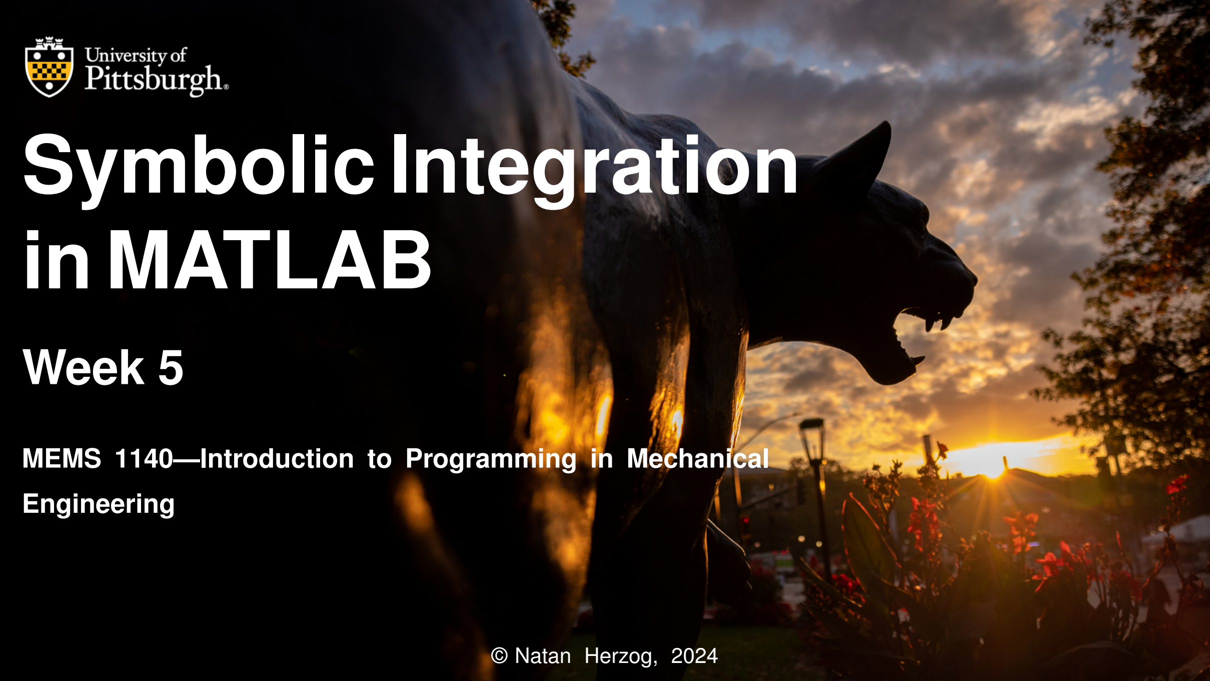
University of Pittsburgh

Symbolic Integration in MATLAB

Week 5

MEMS 1140—Introduction to Programming in Mechanical Engineering



Learning Objectives (L.O.)

At the end of this lecture, you should understand/be able to:

- ❏ What the Symbolic Toolbox is;
- ❏ Define symbolic equations;
- ❏ Solve calculus problems symbolically;
- ❏ Apply symbolic integration to centroid calculations.

Table of Contents (ToC)

1 – What is the Symbolic Toolbox

The Symbolic Toolbox is extremely powerful. It is capable of solving math equations *symbolically*.

In fact, included in the Symbolic Toolbox is the ability to do calculus, including differentiation and integration.

This lecture builds towards evaluating the same centroid from the previous lecture without having to use numeric integration.

2 – Defining Symbolic Equations

To take advantage of the Symbolic Toolbox, a variable has to be specified as a symbol.

This is done with the **syms** command:

```
syms x     % declares that x is a symbolic variable
```

Symbolic variables are used to define symbolic expressions:

```
y = 4*x    % declares a symbolic expressions from the symbolic variable
```

3 – Symbolic Calculus

The Symbolic Toolbox can evaluate calculus like derivatives and integrals for symbolic expressions.

Take the following symbolic expression:

```
>> syms x
>> y = 4*x + 3;
```

Command Window

3 – Symbolic Derivatives

The **`diff`** command can be used to automatically *differentiate* an expression. Its arguments are:

```
diff(<expression>, <differentiation variable>, <order>)
```

```
>> syms x
>> y = 4*x + 3;
>> dy = diff(y, x, 1)    % first-order derivative
dy =
     4
```

Command Window

3 – Symbolic Indefinite Integrals

The **int** command can be used to automatically *integrate* an expression. For ***indefinite*** integration, its arguments are:

```
int(<expression>, <integration variable>)
```

```
>> syms x
>> y = 4*x + 3;
>> dy = int(y, x)        % indefinite integral
dy =
     x*(2*x + 3)
```

Command Window

3 – Symbolic Definite Integrals

The **`int`** command can be used to automatically *integrate* an expression. For ***definite*** integration, its arguments are:

```
int(<expression>, <integration variable>, <lower limit>, <upper limit>)
```

```
>> syms x
>> y = 4*x + 3;
>> dy = int(y, x, 1, 4) % definite integral
dy =
     39
```

Command Window

4 – Application to Centroids

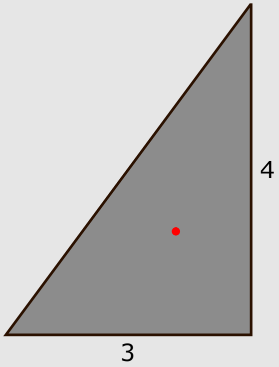
Recall the following equation to evaluate a centroid's location:

$$x_C = \frac{\displaystyle\int_{y_1}^{y_2} x_{C_{dA}} dA}{\displaystyle\int_{y_1}^{y_2} dA}$$

Lecture 4.1 used numeric integration to solve for x_C. Now let's do it using symbolic calculus.

4 – Defining a Symbolic Expression

Define a symbolic expression for $x(y)$, call it **x**:

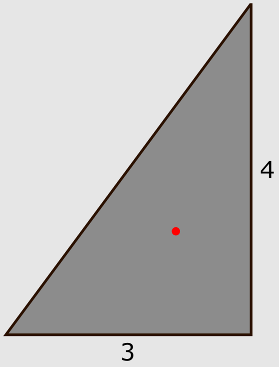


```
syms y
x = 3 - 3/4 * y;
.
.
.
```

4 – Specifying Function Bounds

✓ L.O.1
✓ L.O.2
✓ L.O.3
⇨ L.O.4

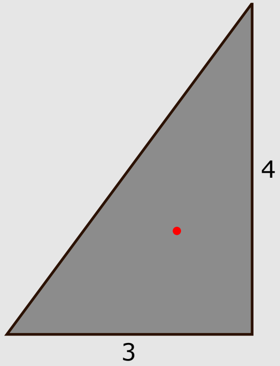
Set the start and end bounds of the function (`y_1` and `y_2`):



```
syms y
x = 3 - 3/4 * y;
y_1 = 0; y_2 = 4;
.
.
```

4 – Numerator of the Centroid

Then the numerator integral is written:



```
syms y
x = 3 - 3/4 * y;
y_1 = 0; y_2 = 4;
x_C = (int(x/2 * x, y, y_1, y_2) / ...
.
```

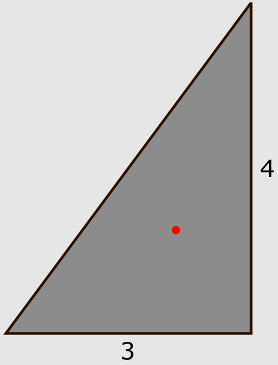
Recall:

$$\int_{y_1}^{y_2} [x_{C_{dA}}(y)]\, dA = \int_{y_1}^{y_2} \left[\frac{x(y)}{2}\right] [x(y)]\, dy$$

4 – Denominator of the Centroid

✓ L.O.1
✓ L.O.2
✓ L.O.3
➪ L.O.4

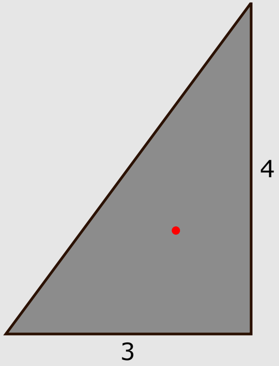
Then the denominator integral is written:



```
syms y
x = 3 - 3/4 * y;
y_1 = 0; y_2 = 4;
x_C = (int(x/2 * x, y, y_1, y_2) / ...
       int(x, y, y_1, y_2))
```

4 – Whole Centroid Calculation

The whole centroid is then determined:



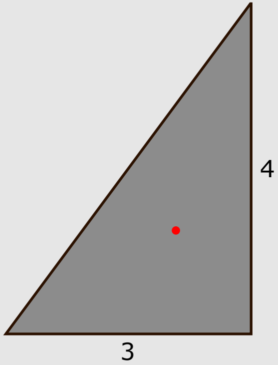
```
syms y
x = 3 - 3/4 * y;
y_1 = 0; y_2 = 4;
x_C = (int(x/2 * x, y, y_1, y_2) / ...
       int(x, y, y_1, y_2))
```

```
x_C =
     1
```

Command Window Output

4 – Correction for Triangle Shape

And recall that **x_C** is taken from the location of the "base:"



```
syms y
x = 3 - 3/4 * y;
y_1 = 0; y_2 = 4;
x_C = 3 - (int(x/2 * x, y, y_1, y_2) / ...
           int(x, y, y_1, y_2))
```

```
x_C =
     2
```

Command Window Output

4 – Final Comparison

Unlike numerical integration, the symbolic result is ***not*** an approximation. It is ***far*** slower than numerics, though.

There is no need to treat the domain as discrete elements.

This eliminates much of the code involved in setting up the domain and evaluating the function at each location.

Ultimately, the Symbolic Toolbox handles the math for you!

5 – Summary

This lecture covered:

✓ What the Symbolic Toolbox is

The Symbolic Toolbox is a powerful part of MATLAB that can solve math in the way that we do, as humans.

✓ How to define symbolic expressions

The **`syms`** command is used to specify variables as symbols, which are then used in defining symbolic expressions. They follow the same form as human-readable math.

5 – Summary

✓ L.O.1
✓ L.O.2
✓ L.O.3
✓ L.O.4

✓ How to solve calculus problems symbolically

Using the **`diff`** and **`int`** commands, symbolic expressions can be automatically differentiated and integrated. The return expressions will also be symbolic expressions.

✓ How to apply symbolic integration to solve for centroids

By employing symbolic integration to the triangle problem from Lecture 4.1, the centroid location can be evaluated with code that more closely resembles human-readable math. And the result is not an approximation.